|  |
| --- |
| **Experiment 1** |
| **Title:** Generate and plot the following signals in time domain and also sketch its amplitude and phase spectrum. Verify the result:   1. Impulse 2. Unit Step 3. Exponential 4. Unit ramp 5. Sinc 6. Rectangular |
| **Learning Objectives** |
| 1. To understand basic standard signals 2. To have hands on simulation using python language |
| **Prerequisites** |
| 1. Basic understanding of mathematics 2. Basic understanding of Python language |
| **Theory** |
| 1. **Unit Impulse Function**   Impulse function is denoted by δ(t). and it is defined as δ(t) = Unit Step FunctionUnit step function is denoted by u(t). It is defined as u(t) =  * It is used as best test signal. * Area under unit step function is unity.  Exponential Signal Exponential signal is in the form of x(t) =  The shape of exponential can be defined by *α* Case i: if *α* = 0 → x(t) = = 1Case ii: if *α* < 0 i.e. -ve then x(t) =. The shape is called decaying exponential.Case iii: if *α* > 0 i.e. +ve then x(t) = . The shape is called raising exponential.Ramp Signal Ramp signal is denoted by r(t), and it is defined as r(t) = Sinc Function It is denoted as sinc(t) and it is defined as sinc(t) == 0 for t=±1, ±2, ±3...Rectangular SignalLet it be denoted as x(t) and it is defined as |
| **Simulation Code** |
| * 1. **Impulse Function**   **Impulse Function**  import numpy as np  import matplotlib.pyplot as plt  # Function to plot Impulse signal d(a)  def unit\_impulse(a, n):  delta = []  for sample in n:  if sample == a:  delta.append(1)  else:  delta.append(0)    return delta    a = 2 # Enter delay or advance  UL = 10  LL = -10  n = np.arange(LL, UL, 1)  d = unit\_impulse(a, n)  #calculating amplitude spectrum usinf fft  y = np.fft.fft(d)  Y\_mag = abs(y) #this gives magnitude of spectrum  plt.subplot(2,2,1)  plt.stem(n, d)  plt.xlabel('n')  plt.xticks(np.arange(LL, UL, 1))  plt.yticks([0, 1])  plt.ylabel('d[n]')  plt.subplot(2,2,2)  plt.xlabel('frequency')  plt.ylabel('Magnitude')  plt.plot(Y\_mag)  plt.subplot(2,2,3)  plt.phase\_spectrum(d) # this shows phase of signal  plt.show()   * 1. **Unit Step**   C:\Users\PRASHIK\AppData\Local\Microsoft\Windows\INetCache\Content.Word\unit_step_simulated.png  import numpy as np  import matplotlib.pyplot as plt    # function to generate unit step u[n-a]  # LL and UL are lower and upper limits of discrete time line  def unit\_step(a, n):  unit =[]  for sample in n:  if sample<a:  unit.append(0)    else:  unit.append(1)  return(unit)    # plot unit step function u[n-a]  a = 1 # Enter delay or advance  UL = 5  LL = -5  n = np.arange(LL, UL, 1)  unit = unit\_step(a, n)  plt.subplot(2,2,1)  plt.stem(n, unit)  plt.xlabel('n')  plt.xticks(np.arange(LL, UL, 1))  plt.yticks([0, 1])  plt.ylabel('u[n]')  plt.title('Unit step u[n-a]')  y = np.fft.fft(unit)  plt.subplot(2,2,2)  plt.xlabel('frequency')  plt.ylabel('Magnitude')  plt.plot(y)  plt.title('Magnitude spectrum')  plt.subplot(2,2,3)  plt.phase\_spectrum(unit)  plt.title('Phase spectrum')  plt.show()   * 1. **Exponential**   **C:\Users\PRASHIK\AppData\Local\Microsoft\Windows\INetCache\Content.Word\exponential_simulated.png**  import numpy as np  import matplotlib.pyplot as plt  sample=[]  a = -1  UL = 1  LL = -1  n = np.arange(LL, UL, 0.1)  N=-n  # Function to generate exponential signals e\*\*(at)  def exponential(a, N):  expo =[]  for sample in n:  expo.append(np.exp( a\*sample))  return (expo)    x = exponential(a, n)  plt.subplot(2,2,1)  plt.stem(n, x)  plt.xlabel('n')  plt.xticks(np.arange(LL, UL, 0.2))  plt.ylabel('x[n]')  plt.title('Exponential Signal e\*\*(an)')  y = np.fft.fft(x)  plt.subplot(2,2,2)  plt.xlabel('frequency')  plt.ylabel('Magnitude')  plt.plot(y)  plt.title('Magnitude spectrum')  plt.subplot(2,2,3)  plt.phase\_spectrum(x)  plt.title('Phase spectrum')  plt.show()   * 1. Unit Ramp   C:\Users\PRASHIK\AppData\Local\Microsoft\Windows\INetCache\Content.Word\ramp_simulated.png  import numpy as np  import matplotlib.pyplot as plt  # Function to generate unit ramp signal r(n)  # r(n)= n for n>= 0, r(n)= 0 otherwise  def unit\_ramp(n):  ramp =[]  for sample in n:  if sample<0:  ramp.append(0)  else:  ramp.append(sample)  return ramp    UL = 5  LL = -5  n = np.arange(LL, UL, 0.5)  r = unit\_ramp(n)  plt.subplot(2,2,1)  plt.stem(n, r)  plt.xlabel('n')  plt.xticks(np.arange(LL, UL, 1))  plt.yticks([0, UL, 1])  plt.ylabel('r[n]')  plt.title('Unit Ramp r[n] in time domain')  plt.subplot(2,2,2)  plt.xlabel('frequency')  plt.ylabel('Magnitude')  plt.magnitude\_spectrum(r)  plt.title('Magnitude spectrum')  plt.subplot(2,2,3)  plt.phase\_spectrum(r)  plt.title('Phase spectrum')  plt.show()   * 1. **Rectangular**   import numpy as np  import matplotlib.pyplot as plt  # LL and UL are lower and upper limits of discrete time line  def rect\_angular(a, T):  mag =[]  UL = T + 5  LL = (-T) - 5  n = np.arange(LL, UL, 1)  for sample in n:  if (sample >= (-T)) and (sample <= T):  mag.append(a)    else:  mag.append(0)  return(mag, n)  a = 1 # Amplitude  T = 5 # Time period  rect, n = rect\_angular(a, T) # calling rect angular fun  plt.subplot(2,2,1)  plt.plot(n, rect)  plt.xlabel('n')  plt.ylabel('u[n]')  plt.title('')  y = np.fft.fft(rect)  plt.subplot(2,2,2)  plt.xlabel('frequency')  plt.ylabel('Magnitude')  plt.plot(y)  plt.title('Magnitude spectrum')  plt.subplot(2,2,3)  plt.phase\_spectrum(rect)  plt.title('Phase spectrum')  plt.show()   * 1. **Sinc**   import matplotlib.pyplot as plt  import numpy as np  Fs = 10  x = np.linspace(-Fs, Fs, 100)  y = np.sinc(x)  plt.subplot(2,2,1)  plt.plot(x, y)  plt.subplot(2,2,2)  plt.magnitude\_spectrum(y)  plt.xlabel('frequency')  plt.ylabel('magnitude')  plt.title('magnitude spectrum')  plt.subplot(2,2,4)  plt.phase\_spectrum(y)  plt.title('Phase spectrum')  plt.show() |
| **Assignment** |
| **Task1: Write description for codes of 2 signals**  **Task2: Vary the parameters of the signals and observe the magnitude and phase spectrum** |